International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 10, Number 2, pages 84–93, July 2020

A Hardware-efficient Weight Sampling Circuit for Bayesian Neural Networks

Yuki Hirayama
Hokkaido University
Sapporo, Hokkaido, 060-0814, Japan

Tetsuya Asai
Hokkaido University
Sapporo, Hokkaido, 060-0814, Japan

Masato Motomura
Tokyo Institute of Technology
Yokohama, Kanagawa, 226-8503, Japan

Shinya Takamaeda
The University of Tokyo
Bunkyo, Tokyo, 113-8656, Japan

Received: February 15, 2020
Revised: April 7, 2020
Accepted: May 11, 2020
Communicated by Ikki Fujiwara

Abstract

The main problems of deep learning are requiring a large amount of data for learning, and prediction with excessive confidence. A Bayesian neural network (BNN), in which a Bayesian approach is incorporated into a neural network (NN), has drawn attention as a method for solving these problems. In a BNN, the probability distribution is assumed for the weight, in contrast to a conventional NN, in which the weight is point estimated. This makes it possible to obtain the prediction as a distribution and to evaluate how uncertain the prediction is. However, a BNN has more computational complexity and a greater number of parameters than an NN. To obtain an inference result as a distribution, a BNN uses weight sampling to generate the respective weight values, and thus, a BNN accelerator requires weight sampling hardware based on a random number generator in addition to the standard components of a deep learning neural network accelerator. Therefore, the throughput of weight sampling must be sufficiently high at a low hardware resource cost. We propose a resource-efficient weight sampling method using inversion transform sampling and a lookup-table (LUT)-based function approximation for hardware implementation of a BNN. Inversion transform sampling simplifies the mechanism of generating a Gaussian random number from a uniform random number provided by a common random number generator, such as a linear feedback shift register. Employing an LUT-based low-bit precision function approximation enables inversion transform sampling to be implemented at a low hardware cost. The evaluation results indicate that this approach effectively reduces the occupied hardware resources while maintaining accuracy and prediction variance equivalent to that with a non-approximated sampling method.

Keywords: Bayesian Neural Network, Lookup Table, Inversion Transform Sampling

1 Introduction

Neural networks require a huge amount of data for learning, are weak against noise, and cannot describe the uncertainty of their predictions. A lack of expression of uncertainty incurs certain disadvantages such as insufficient learning data and an estimation with high confidence even for outliers. Therefore, a Bayesian neural network (BNN), which applies a Bayesian approach to a neural network (NN) is useful. With this method, weights as the parameters of the NN are represented as the distribution, and a prediction is conducted. When obtaining a prediction as a probability distribution, even with a small amount of data, it is possible for a large uncertainty to occur. However, BNNs have a higher computational complexity and a larger number of parameters than NNs based on a point estimation. Thus, in this study, toward the hardware implementation of a BNN, we propose a resource-efficient sampling method and evaluate its performance.
The contributions of this paper are:
- To propose a low hardware overhead weight sampling method for Bayesian neural network accelerators.
- To show that the proposed method achieves the almost equivalent accuracy and variance to that with a non-approximated sampling method.
- To show that the proposed method can be implemented with less hardware resources than the non-approximated method.

2 Bayesian Neural Network

An ordinary NN takes a frequency approach to maximize the likelihood $p(D|w)$ or a posterior probability $p(w|D)$ of the predicted value with respect to the weight w when learning data D are given. For example, in a regression task, the output y maximizes the probability of $y \sim N(f(x, w), \sigma)$. In a classification task, the logarithm of a categorical distribution, that is, the cross entropy is minimized. In this approach, a weight is represented by a single real value, which is point estimated. However, in the Bayesian approach, we assume a probability distribution for the weight, as shown in Figure 1. Thus, the weights are represented as $w \sim p(w)$.

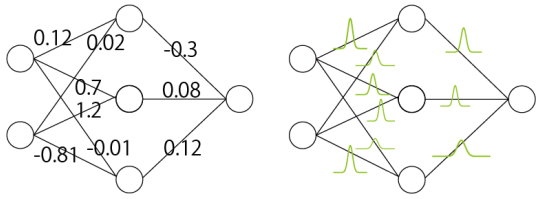

Figure 1: Weight representations of an NN(left) and a BNN(right)

In a Bayesian approach, we use the Bayesian theorem to calculate the posterior distribution. We calculate the posterior distribution $p(w|D)$ of the weight after data D are observed from the likelihood function $p(D|w)$ and prior distribution $p(w)$.

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

This posterior distribution is difficult to calculate directly if the likelihood function and the prior distribution are not conjugated. In practice, a Markov chain Monte Carlo (MCMC) method or variational inference is used. MCMC approximates the unwieldy distribution $p(w|D)$ through sampling. However, there is a problem in that convergence takes time and is not suitable for a large model.

[0]This is an abstract footnote

Variational inference assumes a manageable distribution $q(w|\theta)$ determined by a few parameters instead of an incomputable distribution $p(w|D)$. A parameter θ is found that minimizes the Kullback-Leibler divergence between the two distributions such that the assumed distribution and the true distribution are close to each other[1, 15, 6, 14, 10, 11].

$$\begin{aligned} KL(q(w|\theta)||p(w|D)) &= -\int q(w|\theta) \ln \frac{p(w|D)}{q(w|\theta)} dw \\ &= -\int q(w|\theta) \ln p(D|w) dw + KL(q(w|\theta)||p(w)) \\ &\approx -\frac{1}{N} \sum_{n=1}^{N} \ln p(D|w_n) + KL(q(w|\theta)||p(w)) \end{aligned} \tag{1}$$

Here, N is the number of weight samples, and w_n is the n-th sample. The first term on the right side acts as a data fitting term, and the second term acts as a regularization term. In other words, in variational inference, the calculation of the posterior distribution $p(w|D)$ given learning data D results in finding the distribution $q(w|\theta)$ close to it. To minimize (1), the gradient for the parameter θ is calculated and a gradient descent is conducted. In classification and regression tasks, we want to predict an unknown target value for a new input x. This can be calculated using the variational posterior distribution. Because it is difficult to apply marginalization to all weights, an approximation method is used here as well.

$$\begin{aligned} p(y|x, D) &= \int p(y|w, x) q(w|\theta) dw \\ &\approx \frac{1}{N} \sum_{n=1}^{N} p(y|w_n, x) \end{aligned} \tag{2}$$

2.1 Related Works

C. Blundell et al.[1] proposed the use of Bayes by Backprop, which updates the parameters of the posterior distribution using a backpropagation method to minimize the KL divergence. This makes it possible to calculate the gradient for the KL divergence of the parameters of the posterior distribution and enables a Bayesian treatment of large-scale networks. Here, the parameters of the weight distribution, for example, the gradient of the mean and the variance of the normal distribution, are calculated through backpropagation and updated.

Y. Gal et al.[7] showed that the dropout gives a Bayesian interpretation by assuming a Bernoulli distribution for the weights. In an ordinary dropout, to maintain the generalization, the units are randomly invalidated during learning, and all units are used during an inference. However, because a Bernoulli distribution is assumed for the weight here, a dropout is also applied during the inference and learning stages. Therefore, the predicted distribution can also be derived. When assuming a normal distribution for a weight, it is necessary to define two parameters, μ and σ, for a single weight. In this case, the number of required parameters does not change from a normal NN with the advantages of an uncertainty estimation.

The transition of the number of training data and the accuracy of MNIST[13] are shown in Figure 2 for a general neural network when a Bernoulli distribution is assumed, or when assuming a normal distribution for the weights. When a Bernoulli distribution is assumed, the calculation cost is approximately the same as that of a normal NN, although it is difficult to learn using a dropout, and the accuracy is degraded. In this research, we assume a normal distribution for the weights. Although the numbers of parameters and calculations increase, a more natural and accurate prediction can be made.

Various methods have been proposed for achieving a sampling from the probability distribution during forward propagation. With a reparameterization trick [10], it is possible to update the

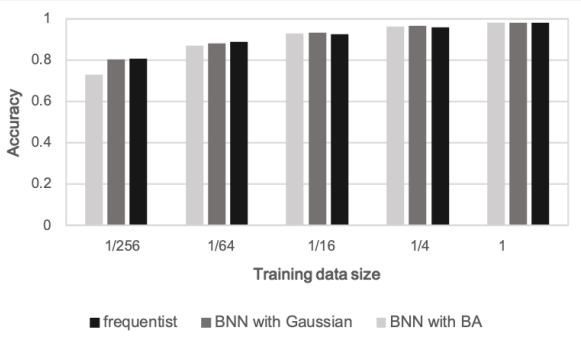

Figure 2: Accuracy on various weight representations

parameters through a gradient descent by obtaining samples of the weights according to an arbitrary distribution by the projection of noise samples. In addition, with a local reparameterization trick [11], it is possible to achieve an inference using fewer samples by sampling not the weight but the activation value, and it is possible to obtain a gradient estimator with a small variance. However, it is necessary to calculate the mean and variance, the calculation cost of which is high.

In addition to an NN incorporating a Bayesian approach, a Bayesian convolutional neural network has also been proposed[6, 14]. A. Kendall et al. [9] introduced random uncertainty arising in the input data and uncertainty caused by the model, and applied them to a computer vision.

Aiming at the hardware implementation of [1], R. Cai et al. [2] proposed an FPGA-based hardware accelerator called VIBNN. In VIBNN, a method to generate normal random numbers by the central limit theorem (CLT) is proposed. In a method based on the CLT, when the binomial distribution B(n, p) has a large n, it approaches a normal distribution to sample normal random numbers. They proposed a RAM-based method to improve the parallelism and scalability, but this has a disadvantage in that the circuit size and power consumption from access to the RAM will increase.

3 LUT-Based Weight Sampling

To reduce the sampling cost in a BNN, we propose a normal random number generation method combining inversion transform sampling and LUT approximation. In an NN, quantization methods that reduce the amount of memory usage and the number of computations are applied by making the values of the weights and activations discrete[3, 8]. Although this method is less accurate than floating-point arithmetic, it is expected to significantly reduce the number of computations. Here, under the hypothesis that the parameters can be quantized in a BNN, we propose sampling from a normal distribution through an inversion transform sampling using a LUT. When assuming a normal distribution for the weights, i.e., $w \sim N(\mu, \sigma^2)$, the mean and variance are required for a single weight. At this time, considering the random variable ϵ with a standard normal distribution as the probability density, an inverse operation of normalization provides weights with an arbitrary mean and variance.

$$w = \mu + \sigma * \epsilon \tag{3}$$

This means that by sampling $\epsilon \sim N(0, 1)$ and conducting an addition using the mean and a multiplication using the standard deviation, random numbers with any normal distribution can be generated.

Sampling methods for normal random numbers include those based on the CLT, the Box-Muller method, and an inversion transform sampling. Because an inversion transform sampling calculates the cumulative distribution of a normal distribution, it is necessary to apply an integral calculation, the calculation cost of which is high. However, considering the effect of quantization in a model

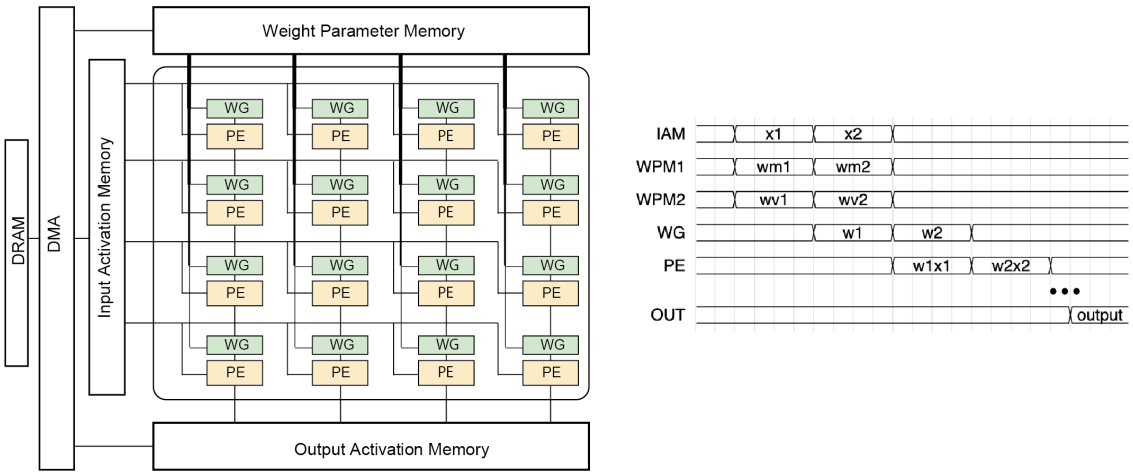

Figure 3: (left) BNN accelerator overview (right) timing chart

weight reduction of an NN, it is considered unnecessary to calculate this strictly. Therefore, using an inversion transform sampling with approximation, sampling from a normal distribution becomes easy.

A simplified accelerator architecture of a BNN and a timing chart are shown in Figure 3. The accelerator consists of a processing element (PE) array, weight generator, and a memory system of weight parameters and input/output activations. Weight parameters such as μ and σ are stored in the weight parameter memory. Here, $\epsilon \sim N(0,1)$ and $w \sim N(\mu, \sigma^2)$ are generated in a weight generator (WG). The procedure consists of (1) reading activation values from the memory to a PE, (2) reading the parameters of weights (mean and variance) from the memory to a PE, (3) sampling (generating) the particular weights from the probability distribution based on the mean and variance, (4) multiplying and accumulating the activations with the sampled weights, (5) applying the activation function, and (6) writing back the calculation results to the memory. These all steps can be implemented in a pipelined manner. The timing chart shows the processing behavior of the PE array described above. Wm1 and wv1 are the mean and variance of the weight w1 respectively. Each multiplication result is obtained from the corresponding weight w1 and input activation x1. This is repeated and accumulated over multiple inputs and weights to obtain the final output. In an NN accelerator, the read weights can be used as is, but in a BNN, it is necessary to calculate the each weight from the read distribution parameters. For high throughput and accuracy, each PE requires a weight generator to sample a distinct weight value, so that reducing the hardware cost of weight generator is certainly effective for its energy-efficiency improvement.

Inverse transform sampling uses an inverse function of the cumulative density function to obtain an output for a sample from a uniform distribution. The inverse function of a standard normal distribution is known as a probit function.

$$F(z) = \int_{\infty}^{z} \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{z^2}{2}} dz \tag{4}$$

However, F^{-1} cannot be obtained analytically, and an approximation method is needed to obtain standard normal random numbers through an inverse transform sampling.

3.1 LUT for Inversion Transform Sampling

Therefore, we propose a sampling using a LUT. To obtain standard normal random numbers through an inversion transform sampling, a table of inverse functions $F^{-1}(u)$ is prepared in advance, and a sample from uniform random numbers is $u \sim U[0,1]$. Here, uniform random numbers are obtained

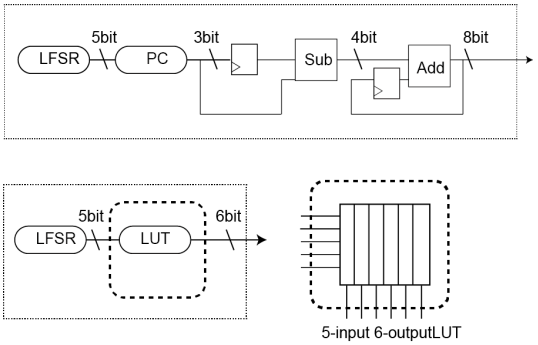

Figure 4: Gaussian random number generator
(top) CLT method (bottom) LUT-based Sampling

using a linear feedback shift register (LFSR). After that, the random number obtained is converted using the table corresponding to the inverse function to obtain the standard normal random number ϵ. It is possible to sample from the desired distribution $N(\mu, \sigma^2)$ by multiplying the ϵ obtained through the standard deviation σ and adding the mean μ. The accuracy of the standard normal random numbers is determined based on the number of entries in the LUT.

Although the number of entries in the LUT can be chosen arbitrarily, the output of the LFSR can be converted into the address of the LUT, given that a decimal fraction with the same accuracy as the bit size can be obtained from the LFSR. However, because the tail of the normal distribution continues indefinitely, we assume a truncated normal distribution here. In this distribution, it is considered that the distribution is limited to a certain range.

The block diagrams of a LUT-based sampling and a CLT method proposed by [2] are shown in Figure 4. We use 5-bits for the input and 6-bits for the output in a LUT. To sample one weight using LUT-based sampling, a random number is sampled from a uniform distribution using an LFSR and transformed into a probit output using a LUT. To obtain an arbitrarily Gaussian random number, it is sufficient to multiply the standard deviation and add the mean weight. When sampling Gaussian random numbers using the CLT of a VIBNN, the binomial distribution is realized using an LFSR, and it is necessary to take the sum of the bit strings. Instead of taking the sum of all bits of the LFSR, the difference is calculated by subtracting the value of the bit string which is one cycle earlier, and the sum is added. Finally, an addition of the mean and a multiplication of the variance are required.

4 Experiments

We evaluated the proposed method's accuracy using several regression and classification tasks and the representation of variance and compared it with the method using ideal Gaussian distribution. Then, we compared the hardware cost of the Gaussian random number generator compared with that of the previous work.

4.1 Regression Tasks

We evaluated the proposed method by using some regression tasks to determine the accuracy and predictive variance. We showed that the proposed method can achieve nearly identical accuracy, and it can express predictive variance well, compared with the Gaussian sampling, which is the ground-truth method. We used four regression datasets from UCI Machine Learning Repository: Boston housing, concrete compressive strength, energy efficiency and yacht hydrodynamics[5]. We used Tensorflow Probability[4] as a framework. The network comprises one hidden layer, which contains 50 neurons. Table 1 shows the root mean square error(RMSE) of these regression tasks.

Table 1: RMSE of regression tasks

dataset	dataset size	RMSE	
		Gaussian	LUT-based sampling
Boston Housing	506	8.93±0.16	9.14±0.25
Concrete Compressive Strength	1030	17.02±0.24	17.33±0.32
Energy Efficiency	768	3.61±0.14	3.66±0.15
Yacht hydrodynamics	308	4.12±0.41	5.18±0.25

In yacht hydrodynamics, the RMSE becomes larger. A possible reason of such the prediction error in the Yacht hydrodynamics task is that the used dataset contains a few training data compared to the datasets of the other tasks. In general, a training dataset is small, the obtained variances of the weights are likely to be large, because there is insufficient information to squeeze the variances. Thus, the harmful effect by the quantization error in the LUT-based approximation has increased. However, with other tasks, LUT-based sampling can achieve nearly identical accuracy compared with ideal sampling.

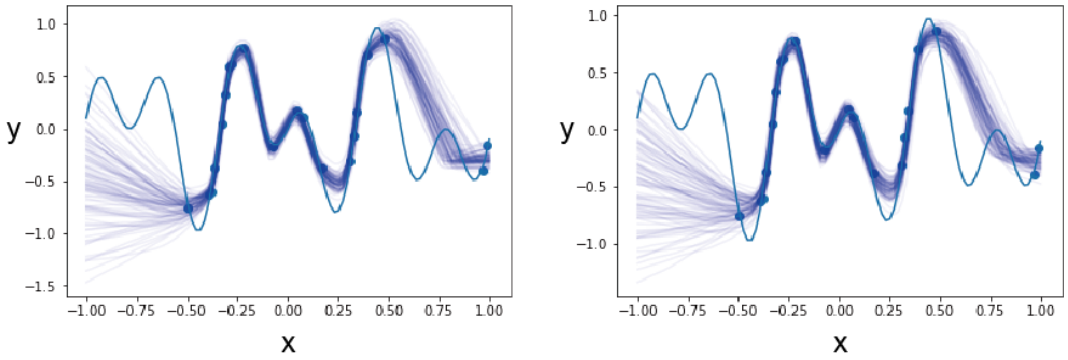

Figure 5: Regression tasks: (left) sampling from a normal distribution (right) LUT-based sampling

We also qualitatively evaluated whether the same variance can be obtained even when using LUT-based sampling. In linear regression tasks, we confirmed the uncertainty of the regression curve when the data points were limited. The true function is set as

$$f(x) = sin(4x) * cos(14x) \tag{5}$$

Let the observation point be $X = \{x_1, ..x_{20}\}$. The corresponding target values $T = \{t_1, ..t_{20}\}$ are f(x) plus Gaussian noise. The network is a fully connected network consisting of 1-1024-512-512-1. The activation function is ReLU. Figure 5 shows the observation point $\{X, T\}$ as the training data and demonstrates the regression curve obtained by sampling the weights multiple times from the posterior distribution of the weights obtained. The regression equation is concentrated in the region where the data points are dense in the sample based on a normal distribution, and the variance is large in a sparse region. In addition, in LUT-based sampling, as in the case of a sample from a normal distribution, it is understood that the predicted variance is large in the region where the data points are sparse. This indicates that, even with the proposed method, when the unknown input x is a sparse region of data points, a prediction with large uncertainty can be made.

4.2 Classification Tasks

We evaluated the accuracy with limited training data using MNIST and CIFAR-10[12]. Examining the accuracy when the training data is limited shows the behavior when the variance of the model is large.

MNIST is a classification task of handwritten digits. There are 60,000 training data points in MNIST, and its accuracy was evaluated using 10,000 evaluation data points. We divided the training data into 1/4, 1/16, and 1/256 parts for evaluation. The accuracy was evaluated when sampling from a Gaussian distribution without using an approximation, and when sampling through the inverse function method using a LUT. We assumed a standard normal distribution as a prior distribution, normal distribution with the mean μ and standard deviation σ as a posterior distribution. We used the following network for MNIST:

$$784(= 28 * 28) - 200 - 200 - 10$$

We use the following network for CIFAR-10:

$$3072(= 32 * 32 * 3) - 64 - 64 - 128 - 128 - 128 - 128 - 256 - 10$$

The reason for using a fully connected neural network instead of a convolutional neural network is that a hardware implementation of Bayesian CNN has not been proposed yet.

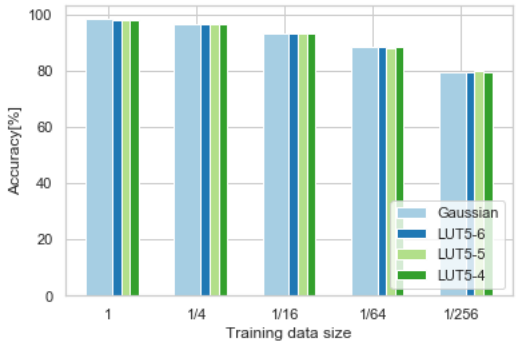

Figure 6: Accuracy on the MNIST datasets

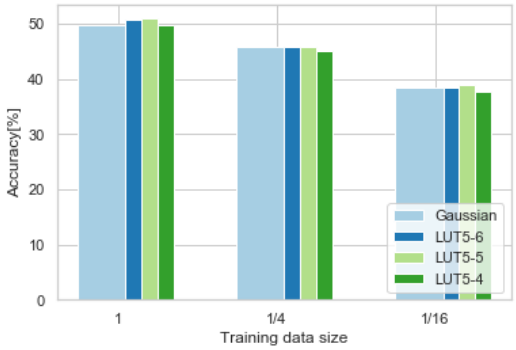

Figure 7: Accuracy on the CIFAR-10 datasets

Figure 6 and Figure 7 show the accuracy when ten inferences are made. The accuracy is calculated using the average value of predictions obtained by sampling ten times. LUT5-6 means using 5-bit input and 6-bit output LUT for weight sampling. From Figure 6 and Figure 7, it can be seen that the accuracy decreases as the training data becomes smaller in each method. However, regardless of the training data size, the difference in accuracy between the methods is small and LUT-based sampling performs as well as Gaussian sampling. This indicates that the effect of weight quantization is small even when the size of training data is small. One reason is that the effect of the central limit theorem. As the addition of random variables increases, the output is expected to follow a normal distribution. Therefore, the output depends on the mean and variance of the input, so that it is considered that the influence of the quantization is small.
The LUT-based sampling achieves higher accuracy than Gaussian sampling in CIFAR-10. This is considered that Gaussian sampling generates a weight value from a wide numerical range strictly, even if the trained variance is sufficiently small. In contrast, LUT-based sampling can obtain a stable weight value from a confident and narrow numerical range, when the trained variance is small. In other words, a stable inference can be performed even when the number of MC sampling is small in point of accuracy.

4.3 Hardware Resources

We compared the proposed LUT-based sampling with a VIBNN from a previous study. We confirmed the necessary hardware resources of the proposed method and the CLT method using an Artix-7

Table 2: Harware Usage of GRNG

method	This work	RLF-GRNG[2]
Slice LUTs	3	12
Slice Registers	0	11

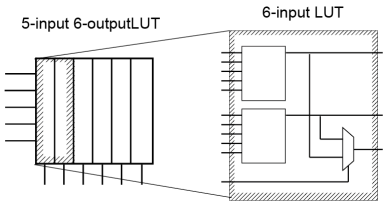

Figure 8: Target LUT realized by multiple FPGA LUTs

XC7A35T-L1CSG324I as the target device. We use Xilinx vivado v2018.3 as a synthesis tool. The hardware resources usage for each method is shown in Table 2.
In the evaluation, we used a 5-bit input and 6-bit output LUT for sampling. In an FPGA, a configurable logic block (CLB) consists of LUTs and registers. In a 7-Series FPGA developed by Xilinx, a 6-bit input LUT can implement any Boolean function and can be used as two 5-bit input LUTs if they share inputs as shown in Figure 8. For this reason, if a 5-bit input, 6-bit output LUT is used for LUT-based sampling, a Boolean function can be realized using three LUTs. Thus, the LUT-based sampling can be conducted using only three 6-bit input LUTs (= six 5-bit input LUTs), and no registers. DNN accelerators generally consist of a large number of processing units. For BNN, each unit requires a random number generator to obtain independent weight. Therefore more resource efficient BNN accelerators can be realized by employing the proposed method.

5 Conclusion

In this paper, we focused on a newly proposed method for obtaining random numbers according to the desired distribution from a single random number through LUT, which we call LUT-based sampling. The results of the evaluation show that, even with LUT-based sampling, an accuracy equivalent to a sample from a Gaussian distribution can be obtained regardless of the size of the dataset. Moreover, although only a quantitative evaluation was conducted, for a regression task, it was shown that a predicted variance similar to the sample from a Gaussian distribution can be obtained by visualizing the variance. Finally, hardware resources were evaluated using a method based on the CLT of a VIBNN, which is a previous approach, and through hardware implementation. The synthesis result indicates that the hardware resources per GRNG to obtain Gaussian random numbers from uniform random numbers are reduced. As a future work, we plan to apply the

proposed approach to convolutional neural networks incorporating a Bayesian method. Moreover, although the sampling cost of a BNN can be lowered, it is necessary to make multiple inferences for quantifying the uncertainty, and we believe that there is room for improvement in this respect.

References

[1] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1613–1622, Lille, France, 07–09 Jul 2015. PMLR.

[2] Ruizhe Cai, Ao Ren, Ning Liu, Caiwen Ding, Luhao Wang, Xuehai Qian, Massoud Pedram, and Yanzhi Wang. VIBNN: hardware acceleration of bayesian neural networks. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2018, Williamsburg, VA, USA, March 24-28, 2018*, pages 476–488, 2018.

[3] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3123–31,31. Curran Associates, Inc., 2015.

[4] Joshua V. Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matthew D. Hoffman, and Rif A. Saurous. Tensorflow distributions. *CoRR*, abs/1711.10604, 2017.

[5] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[6] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *4th International Conference on Learning Representations (ICLR) workshop track*, 2016.

[7] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[8] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4107–4115. Curran Associates, Inc., 2016.

[9] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5574–5584. Curran Associates, Inc., 2017.

[10] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[11] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc., 2015.

[12] Krizhevsky, Alex, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[13] Yann LeCun and Corinna Cortes. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.

[14] Kumar Shridhar, Felix Laumann, and Marcus Liwicki. Uncertainty Estimations by Softplus normalization in Bayesian Convolutional Neural Networks with Variational Inference. *arXiv e-prints*, page arXiv:1806.05978, Jun 2018.

[15] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. FUNCTIONAL VARIATIONAL BAYESIAN NEURAL NETWORKS. In *International Conference on Learning Representations*, 2019.